Problema 5.1.

Codifica un programa PASCAL que calcule el factorial de un número entero positivo, comprendido entre 0−7. El valor se introduce por teclado y se muestra el resultado por pantalla. Define un subprograma recursivo.

NOTA: Factorial de 0 es 1

Ejemplos de entradas:

- Introduce Número (0−7): 2
- Introduce Número (0−7): −2
- Introduce Número (0−7): 8

Ejemplos de Salidas correspondientes:

- El factorial de 2 es 4
- El número debe ser positivo

Introduce Número (0−7):

- El número debe estar en el intervalo [0,7]

Introduce Número (0−7):

Solución

program **FactorialRecursivo** (input,output); {Calcula el factorial de un número entero positivo de forma recursiva }uses crt;var x, f:integer; procedure **TomaDato** (var x:integer); {Postcondicion: Input=n y n>=0 y x=n }

begin

{Precondicion: −−}

repeat

write ('Introduce Número Entero Positivo (0−7): ');

readln (x); {Asegura la salida del bucle}

if x<0 then

writeln ('El número ha de ser entero positivo')

else

if x>7 then

writeln ('El número ha de ser menor que 8');

writeln;

until (x>=0) and (x<=7);

{Poscondicion del ciclo:Input=n y 0<=n<=7 y x=n }

end; {de TomaDato}

function **Factorial** (x:integer):integer; {Precondición: 0<=x<=7 Postcondicion: factorial= x *(x−1)!" factorial=1 } begin if x=1 then Factorial:=1 {Caso Base} else Factorial:= x * Factorial(x−1); {Ley de Recurrencia}

end; {de factorial}

begin {Programa Principal}

clrscr;

TomaDato (x);

f:=Factorial(x);

writeln (x,'!=',F,' El factorial de ',x,' = ',F);

end. {del Programa Principal}

Problema 5.2.

Codifica un programa PASCAL que admita caracteres por teclado hasta pulsar Ctrl−Z (eof), y muestre por pantalla los caracteres introducidos en orden inverso a como se introdujeron. No se puede utilizar arrays. Utilizar un subprograma recursivo.

program **InvierteCaracteres** (input,output);

{Admite caracteres hasa pulsar (ctrl−z) y los muestra invertidos}

procedure **Admite**; var x:char; begin {Caso Base: eof} if not eof then begin read (x); {Ley de Recurrencia} admite; write(x)

end

end;

begin {Programa Principal}

writeln ('Escribe Caracteres y te los mostrará en pantalla al revés.

Termina con ctrl−z');

Admite;

end.

Problema 5.3.

Utiliza subprogramación recursiva para escribir un programa en PASCAL que admita por teclado un real (n) y un entero (e) y muestre por pantalla la potencia e de n.

program **PotenciaRecursiva** (input,output); {Acepta n y e por teclado y devuelve n elevado a e. Utiliza subprograma recursivo} var n, resultado:real;

e:integer;

procedure **TomaDatos(**var n:real;var e:integer); begin writeln; write('Introduzca base: '); read(n) ; write('Introduzca el exponente: '); read(e) ; end;

function **Eleva** (n:real;e:integer):real;

function **ElevaExppos** (n:real;e:integer):real;

{Precondición e>=0

Postcondición ElevaExppos=n * ElevaExppos(n,e−1) " 1}

begin

if e=0 then

elevaexppos:=1 {Caso Base}

else

elevaexppos:=n * elevaexppos (n,e−1);

{Ley de Recurrencia}

end; {De elevaexppos}

begin {De eleva}

case e of

−99..−1:eleva:=1/elevaexppos(n,abs(e));

0..99:eleva:=elevaexppos(n,e);

end;

end; {de eleva}

begin **{Programa Principal}**

TomaDatos(n,e);

resultado:=Eleva(n,e);

```
writeln (n:5:2,' ** ',e, ' es ',resultado:5:2);

end. {del Programa Principal}
```

Problema 5.4.

Utiliza subprogramación recursiva para escribir un programa en PASCAL que admita por teclado un entero positivo (n) muestre por pantalla el valor de succesión Fibonacci que le coresponde:

Sucesión Fibonacci: 1,1,2,3,5,8,13,24,

Fibo (0) = 1

Fibo (1) = 1

Fibo (6) = 13

```
program Fibonacci (input,output);

var

x:integer;

procedure TomaDato (var x:integer);

{Postcondición: Input=n y n>=0 y x=n }

begin

x:=0;

repeat

if x<0 then

writeln ('El n£mero ha de ser entero positivo');

writeln;

write ('Introduce Número Entero Positivo: ');

readln (x) {Asegura la salida del bucle}

until (x>=0);

end; {de TomaDato}

function Fibo (x:integer):integer;

{Precondicion: x>=0}

begin
```

```
if (x=0) or (x=1) then

Fibo:=1 {Caso Base}

else

Fibo:=Fibo(x−2)+Fibo(x−1); {Ley de Recurrencia}

end;

begin {Programa Principal}

TomaDato(x);

writeln ('Fibonacci de ',x, ' es ', Fibo(x));

end.
```

Problema 5.5.

Utiliza subprogramación recursiva para escribir un programa en PASCAL que solucione el problema de las Torres de Hanoi

```
program Hanoi (input,output);

const

Torre1='A';

Torre2='B';

Torre3='C';

var

n:integer; {número de discos}

procedure Pasar (n:integer;Ini,Fin,Aux:char);

{Escribe la secuencia de movimientos con tres torres y n anillos}

{Precondicion: n>=0 }

begin

{Caso Base n=0}

if n>0 then

begin {Ley de Recurrencia}

Pasar (n−1,Ini,Aux,Fin);
```

```
writeln('Mover disco ',n:3, ' desde ',ini:3, ' a', fin:3);

Pasar (n−1,Aux,Fin,Ini)

end

end; {de Pasar}

begin {Programa Principal}

write ('Introduce n£mero de discos: ');

readln(n);

Pasar(n,Torre1,Torre2,Torre3);

end.
```

Problema 5.6.

Escribe un subprograma versión recursiva del MCD de dos números enteros por el método de Euclides. Estudia y compara la solución iterativa con la solución recursiva.

```
program CalculaMCD (input,output);

{Calcula el máximo común divisor de dos n£meros enteros, positivos,

y distintos tomados por teclado. Utiliza una versión Recursiva}

uses

crt;

var {Globales}

x1,x2,MCD:integer;

procedure TomaDatos (var x1,x2:integer);

{Postcondicion: Input=[m,n], m>0 y n>0, x e y >0}

begin

repeat

writeln;

write('Introduzca el entero: ');

read(x1) ;

write('Introduzca el entero: ');
```

```
read(x2) ;

until (x1>0)and( x2>0)

end;

function Calcula (x1,x2:integer):integer;

{Precondición de Calcula: x1 >0 y x2 > 0}

begin

if x2=0 then

Calcula:=x1 {Caso Base}

else

calcula:=calcula(x2, x1 mod x2) {Ley de Recurrencia}

end;

begin {Programa Principal}

clrscr;

TomaDatos(x1,x2);

MCD:=Calcula(x1,x2);

writeln (' El MCD es: ',MCD);

end. {del Programa Principal}
```

Problema 5.7.

Escribe un subprograma versión recursiva que determine la paridad de un entrero positivo introducido por teclado. Estudia y compara la solución no recursiva con la solución recursiva.

```
program ParImpar (input,output);

var

n:integer;

function EsImpar(n:integer):boolean; forward; {Predeclaración}

function EsPar(n:integer):boolean;

begin

if n=0
```

```
then EsPar:=true {Caso base}

else

EsPar:=EsImpar(n−1); {Ley de recurrencia}

end;

function EsImPar(n:integer):boolean;

begin

if n=0

then EsImPar:=false {caso base}

else

EsImPar:=EsPar(n−1); {Ley de recurrencia}

end;

begin {Programa Principal}

writeln (`Introduce n: ');

readln (n);

writeln(EsPar(n));

end.
```

Problema 5.8.

Escribe un subprograma versión recursiva que admita una secuencia de caracteres con paréntesis y corchetes y muestre por pantalla paso a paso el equilibrado de paréntesis y corchetes.

```
program ParentesisCorchetes (input ,output);

{Estudia el equilibrado de paréntesis y corchetes en secuencia de

caracteres}

var

c:char;

procedure CierraCorchete;forward; {Predeclaración}

procedure CierraParentesis;

{Precondicion: Se ha leido un caracter '(' y no es eoln}
```

```
{Postcondicion: Se ha recorrido la entrada hasta encontrar un carácter
')' o el fin de la entrada, dando los mensajes
adecuados si se ha leído un símbolo inadecuado}
var
c:char;
begin
repeat
read (c);
case c of
'(':CierraParentesis;
')':writeln ('Cuadra Par ntesis');
'[':CierraCorchete;
']':writeln ('Error: intentas cerrar un paréntesis con
corchete');
end; {del case}
until (c=')') or eoln;
if eoln and (c<> ')') then
{Se llega al final de la entrada sin encontrar el caracter ')'}
writeln ('ojo: El par ntesis se queda abierto');
end;{de CierraParénesis}
procedure CierraCorchete;
{Precondicion: Se ha leido un caracter '[' y no eoln}
{Postcondicion: Se ha recorrido la entrada hasta encontrar un
caracter ']' o el fin de la entrada, dando los
mensajes adecuados si se ha le¡do un símbolo
inadecuado}
```

```
var
c:char;
begin
repeat
read (c);
case c of
'(':CierraParentesis;
')':writeln ('Error: intentas cerrar un corchete con
paréntesis');
'[':CierraCorchete;
']':writeln ('Cuadra corchete');
end; {del case}
until (c=']') or eoln;
if eoln and (c<> ']') then
{Se llega al final de la entrada sin encontrar el caracter ')'}
writeln ('ojo: El corchete se queda abierto');
end;
begin {de ParentesisCorchetes}
repeat
read(c);
case c of
'(': if not eoln then
CierraParentesis
else
{La entrada acaba en '('}
writeln ('Error: Se queda un par ntesis abierto');
```

```
')':writeln('Error: par ntesis cerrado inadecuadamente');

'[':if not eoln then

CierraCorchete

else

writeln ('Error: El corchete queda abierto');

']':writeln ('Error: corchete cerrado inadecuadamente')

end; {del case}

until eoln;

end.
```